Analysis and Discovery of Service Orchestrations

Carlos M. D. Libório
Supervisor: Diogo R. Ferreira

IST - Technical University of Lisbon

Abstract. Service orchestrations are used by organizations to develop, implement and deploy business processes based on the linkage, composition and reutilization of services. These service orchestrations are modelled using a workflow technology prescribing service behaviour and interactions, and constitute themselves reusable services. Such services are used to implement organizational processes and can be invoked by other organizations, enabling business networkability, organization interoperability, and service globalization.

This work aims at analysing and discovering service orchestrations by means of applying process mining techniques. This way, it will be possible to analyse the run-time behaviour of service orchestrations, extract their run-time process models and compare them with the original models in order to improve or redesign business processes for an effective use of services in an enterprise environment.

Keywords: Service Orchestrations, Process Mining, Service Oriented Architectures, Interoperability, Business Processes

1 Introduction

Today, information systems are at the core of any organization, allowing, not only better adaptability but also interoperability. Globalization reinforces the need of organizations to perform better, faster, more efficiently, but also, the urge to communicate and develop partnerships [10,11]. Business processes are no longer a one communication node only, they have the ability to “travel” through several organizations, becoming cross-organizational processes.

Service oriented architectures allow organizations to implement such business processes in a flexible way, modelling the behaviour of service orchestrations as workflows [20,12]. These service orchestrations represent or model the implementation of one or more business processes along the organization’s service and application portfolio. After deployment and during their execution time, service orchestrations generate vast amounts of run-time information that can be used to monitor and analyse an organization’s business processes. This work introduces a process mining methodology to analyse and discover deployed service orchestrations using their execution generated event logs, with the purpose of studying and unveiling run-time behaviours and performance issues. A methodology for the analysis and discovery of service orchestrations (ADSO) is proposed.
and discussed, along with a software application that implements such mining methodology, and a case study built, in which developed techniques are validated using one integration platform, but application is also possible in other integration scenarios.

This work is organized as follows. Service orchestrations are introduced and discussed in section 1. Section 2 refers to process mining, its goals and how it can be useful regarding the analysis and discovery of service orchestrations. A proposed process mining methodology and solution is addressed in section 3. Section 4 presents a service orchestrations’ mining application that applies the proposed mining methodology. A case study is performed, where a service orchestration is used as an application example, so as to validate the proposed solution and test the mining application, in section 5. Finally, conclusions are drawn in section 6.

2 Service Orchestrations

According to [161814137], a service orchestration is a recursive composition of services, in the sense that a service is built upon existing services. Activity and message flow are controlled by an orchestration service, also encapsulating workflow logic, the service logic. This can be depicted in figure 1, where message exchanges and business logic is present. WS-BPEL[1] is the standard for orchestration language definition [11512196].

![Service orchestration as a workflow logic with service composition](image)

Service orchestrations are used to compose and create services that belong to a specific process participant, describing how services can interact with each other at the message level, and the execution order of such interactions, thus

creating business processes from the composition of services. Using service orchestrations, it is possible to develop, deploy and execute business processes that interact with external services. Enabling service and process collaboration, and therefore providing potential integration endpoints for other processes or services, service orchestrations become integration enablers.

Figure 2 illustrates a service orchestration, designed using Microsoft Biztalk integration server.

![Service orchestration design in Microsoft Biztalk integration platform](image)

As one can observe, there is an activity flow with message exchanges and some service logic. The process model depicted in figure 2 is easily understood, starting by receiving a request message, in the “ReceiveRequest” activity, through some communication port. This message contains an order request with the quantity for some product. The “CheckQuantity” activity is no more than a business rule that simply investigates this quantity and follows some rule. If the request is accepted, the request is sent to the ERP system through the “SendReqToERP” activity. Otherwise, a request denied message is constructed and sent back using some other communication port.

Service orchestrations are centralized services, thus potentially becoming a bottleneck, but they can be deployed as web services or communicate with other orchestrations, which can implement a more distributed service orchestration, enabling scalability and improving performance; "Through the use of orchestrations, service-oriented solution environments become inherently extensible and
adaptive, and, for many environments, orchestrations become the heart of SOA” [9].

3 Process Mining

Process mining is the retrieval of “hidden” knowledge about a process or service, under some perspectives, improving the process model itself and its awareness. Its goal is the automated process information discovery from a process event log [34]. Any information systems such as Customer Relationship Management (CRM), Workflow Management (WfMS), Enterprise Resource Planning (ERP), or Business-to-Business (B2B) systems, have some kind of event log, sometimes referred to as “history”, “audit trail”, or “transaction log”. Often these logs are placed in a single file, several files, or databases.

Table 1 shows the process log retrieved from the deployed service orchestration presented in section 2; (i) Process instance refers to each orchestration’s case or instance; (ii) Activity refers to each task or instruction. In the case of service orchestrations, these tasks are mainly system activities, decision points or communication ports, but there can be human interaction activities also; (iii) The event type represents the transactional model of each event, and is useful to service orchestrations when dealing with parallelism, since information about the beginning and ending of each activity is available, as start and complete event types (iv) Originator gives reference to whom has started or executed the

<table>
<thead>
<tr>
<th>Process Instance</th>
<th>Activity Name</th>
<th>Event Type</th>
<th>Originator</th>
<th>Timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Initialization</td>
<td>start</td>
<td>...</td>
<td>28-10-2009 19:33:51.293</td>
</tr>
<tr>
<td>1</td>
<td>ReceiveRequest</td>
<td>start</td>
<td>...</td>
<td>28-10-2009 19:33:51.357</td>
</tr>
<tr>
<td>2</td>
<td>Initialization</td>
<td>start</td>
<td>...</td>
<td>28-10-2009 19:33:51.370</td>
</tr>
<tr>
<td>2</td>
<td>ReceiveRequest</td>
<td>start</td>
<td>...</td>
<td>28-10-2009 19:33:51.370</td>
</tr>
<tr>
<td>2</td>
<td>ReceiveRequest</td>
<td>complete</td>
<td>...</td>
<td>28-10-2009 19:33:51.370</td>
</tr>
<tr>
<td>2</td>
<td>CheckQuantity</td>
<td>start</td>
<td>...</td>
<td>28-10-2009 19:33:51.370</td>
</tr>
<tr>
<td>2</td>
<td>ConstructRequestDenied</td>
<td>start</td>
<td>...</td>
<td>28-10-2009 19:33:51.370</td>
</tr>
<tr>
<td>1</td>
<td>ReceiveRequest</td>
<td>complete</td>
<td>...</td>
<td>28-10-2009 19:33:51.573</td>
</tr>
<tr>
<td>1</td>
<td>CheckQuantity</td>
<td>start</td>
<td>...</td>
<td>28-10-2009 19:33:51.573</td>
</tr>
<tr>
<td>1</td>
<td>SendReqToERP</td>
<td>start</td>
<td>...</td>
<td>28-10-2009 19:33:51.590</td>
</tr>
<tr>
<td>2</td>
<td>ConstructRequestDenied</td>
<td>complete</td>
<td>...</td>
<td>28-10-2009 19:33:51.600</td>
</tr>
<tr>
<td>2</td>
<td>SendReqDenied</td>
<td>start</td>
<td>...</td>
<td>28-10-2009 19:33:51.600</td>
</tr>
<tr>
<td>2</td>
<td>SendReqDenied</td>
<td>complete</td>
<td>...</td>
<td>28-10-2009 19:33:51.600</td>
</tr>
<tr>
<td>2</td>
<td>CheckQuantity</td>
<td>complete</td>
<td>...</td>
<td>28-10-2009 19:33:51.600</td>
</tr>
<tr>
<td>2</td>
<td>Initialization</td>
<td>complete</td>
<td>...</td>
<td>28-10-2009 19:33:51.600</td>
</tr>
<tr>
<td>1</td>
<td>SendReqToERP</td>
<td>complete</td>
<td>...</td>
<td>28-10-2009 19:33:51.620</td>
</tr>
<tr>
<td>1</td>
<td>CheckQuantity</td>
<td>complete</td>
<td>...</td>
<td>28-10-2009 19:33:51.620</td>
</tr>
<tr>
<td>1</td>
<td>Initialization</td>
<td>complete</td>
<td>...</td>
<td>28-10-2009 19:33:51.620</td>
</tr>
</tbody>
</table>

Table 1. Service orchestration’s event log
activity, the performer; (v) Timestamp is the execution time of each activity. Event logs such as the one presented in table 1 are the starting point for process mining [34].

Considering service orchestrations, these are some of the relevant questions that can be answered using process mining techniques:

1. How is the service orchestration modelled? What are the business rules? Are they coherent and executed correctly?
2. How was the orchestration built? How many versions does it have? Can one see its growth along time?
3. What are the most frequent paths and their execution probabilities along the orchestration?
4. What is the average/maximum/minimum performance time for each activity and for the whole orchestration? And for external services, whenever present?
5. Is the process model efficient and adequate for its purpose? Are there any bottlenecks? What are the critical paths?
6. How does the service orchestration conform with the previous designed model?
7. What are the external service communications?
8. What is the service architecture in the organization?

The following section will present the proposed solution based on a process mining methodology, with the purpose of analysing event logs from deployed orchestrations and discovering their run-time behaviours, unveiling process-aware and performance information that will help to tackle these questions, and find some meaningful answers.

4 Mining Service Orchestrations

The proposed solution is focused mainly on the control-flow, or process perspective, since the main concern is in gathering the orchestration’s design model. Besides the logical model, some performance issues are also of interest. Figure 3 illustrates the main steps involving the proposed ADSO process mining methodology, where ADSO stands for “Analysis and Discovery of Service Orchestrations”.

![Fig. 3. Analysis and discovery of service orchestrations (ADSO), the mining methodology](image)

To apply the depicted ADSO mining methodology, the following phases are presented [15]:

1. Log Extraction
2. Log Inspection
3. Process Model
4. Model Visualization
5. Performance Analysis
6. Conformance Analysis
7. Service Network Architecture
1. Log extraction: Identify every relevant information and where it is stored by the integration platform.

2. Log inspection: Event logs are retrieved and pre-processed. Errors or some inconsistencies are checked and unnecessary information is removed. This step is also useful when applying filters, like choosing process instances by date or by deployment version.

3. Log aggregation: The ordering of activities is retrieved, and the executed logical flow modelled. There are two possible ways here, namely:
   (a) MXML log file: The pre-processed log file is mapped to Mining-XML format. Considering the event types found in the logs, only two types from the transactional model [8] were chosen, start and complete.
   (b) Process model: Using a control-flow perspective, data is mined and the orchestration’s design model is created. A dependency frequency table [20,21] is built to achieve this step.

4. ProM analysis: Process mining analysis can be made in the ProM framework, using the previous MXML log file.

5. Model Visualization: The orchestration’s design model can be visualized, by means of a dependency graph.

6. Performance analysis: Activity and process performance can be analysed, regarding execution times and path probabilities.

7. Conformance analysis: The execution or run-time behaviour of the mined orchestration can be compared with the idealized and initially designed model. Both models can be matched together, showing any existing discrepancies between the initial process model and the deployed process execution [17].

8. Service network architecture: Service communications are analysed and reported, which can be used to discover the wider service architecture.

All phases involved in the ADSO methodology can be seen as the *breadcrumbs* to discover process-aware information about service orchestrations. OrchInsider, a service orchestrations mining application that applies the presented solution, will be introduced in the following section.

5 OrchInsider: a Service Orchestrations’ Mining Application

A software application, named as OrchInsider, was developed and designed to deliver the ADSO mining methodology, using Microsoft Biztalk integration server. Although it was initially conceived to support this integration platform system, OrchInsider can easily be adapted to support other integration platforms, only requiring the capability of extracting event logs from other systems. This can be accomplished by adding a software “plugin”, or by using the MXML data log format. Also, OrchInsider can work together with ProM, the process mining framework, since it can export discovered orchestration event log files to MXML. Nevertheless, more specific orchestration analysis is not provided by the ProM
framework, like version analysis, communication analysis or even conformance analysis. OrchInsider has been developed to analyse and discover the following aspects of deployed service orchestrations:

- Version Analysis: Observe how the logical model was developed and evolved during the course of time.
- Communication Ports: Understand the orchestration’s communication behaviour and depict communication architecture.
- Design Model: Visualize the orchestration’s design model, the process logic.
- Bottleneck Analysis: Configure bottleneck thresholds and analyse bottleneck activities.
- Path Analysis: Reveals correctly and non-correctly executed orchestrations.
- Conformance Analysis: Compares initially designed models with mined models.

Figure 4 displays the orchestration’s design model, enhanced with performance analysis, for the orchestration presented in section 2.

Fig. 4. Service orchestration’s analysis design model, using OrchInsider

The presented model depicts every behaviour “learned” from the event log shown in table 1. Activities are pictured as rounded boxes and business decision

2ProM’s conformance plugins can only be applied if parsed service orchestration models are exported.
rules as diamond boxes, and every activity is drawn with a gray tonality according to its execution time and related to the orchestration’s execution time. It is easy to find some similarities when comparing figure 4 with figure 2. All orchestration instances and versions were selected and mined. When mining service orchestrations, several options can be selected using OrchInsider. For instance, one could define a specific list of instances to analyse, or select instances between start and end dates, and even target results by selecting which version/s to analyse. In the following section, a brief case study will be presented were this mining tool is validated, and tested with a more complex service orchestration.

6 Case Study: SportTicket Online

SportTicket Online was an academic project from the Enterprise Systems Integration course at the Technical University of Lisbon (IST). This project consists of relatively complex service orchestrations that integrate several services to build a ticket reseller for the World Cup Football championship 2010. The main purpose of the project is to manage ticket reselling for the world cup, supporting the buying and also the returning processes, both implemented with service orchestrations.

This case study will provide a brief analysis of the buying process, using OrchInsider, so it is possible to perceive the possibilities of a more profound analysis. SportTicket Online’s buying process consists of several interconnected steps, since the game and ticket selection, pricing rules, customer data verification, payment, billing, and ticket emission. Loading this orchestration into OrchInsider, one can check that there were multiple orchestration instances executed, and three orchestration versions are present.

Using OrchInsider, it is possible to visualize the orchestrations’s design model and enrich such model with more information that can help to characterize mined behaviour. To better realize activity performance and path frequency, the analyst can define a bottleneck threshold, according to defined process expectations, and visualize the mined model with identified bottleneck activities and path frequency information, as can be observed in figures 6, 7, and 8. Performance information can this way be easily attained, which helps to quantify and improve mined orchestration’s execution characterization.

Figure 5 presents SportTicket Online’s buying process performance information, retrieved from OrchInsider’s performance analysis.

<table>
<thead>
<tr>
<th>Orchestration Performance Info</th>
<th>Max Time</th>
<th>Min Time</th>
<th>Average Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orchestration Time</td>
<td>00:00:16.967</td>
<td>00:00:05.310</td>
<td>00:00:12.641</td>
</tr>
<tr>
<td>Internal Activities</td>
<td>00:00:11.000</td>
<td>00:00:05.394</td>
<td>00:00:08.748</td>
</tr>
<tr>
<td>External Activities</td>
<td>00:00:19.961</td>
<td>00:00:05.992</td>
<td>00:00:12.948</td>
</tr>
</tbody>
</table>

Fig. 5. SportTicket Online’s buying process’ performance information
Fig. 6. *SportTicket Online’s* mined buying process enhanced with performance analysis (a)
Fig. 7. *SportTicket Online’s* mined buying process enhanced with performance analysis (b)
Fig. 8. SportTicket Online’s mined buying process enhanced with performance analysis (c)
Considering maximum execution times, one can observe that the orchestra-

tion’s execution time was 1 minute, 48 seconds and 987 milliseconds; Internal

activities represent 0:11.006 whereas external activities stand for 1:37.981 of the

total execution time. So, almost all of the mined orchestration’s execution time

is consumed “outside” the orchestration, in invoked external services.

OrchInsider’s path analysis is helpful to characterize orchestration execution,

understanding if terminated instances occurred. A terminated instance repre-

sents an orchestration’s execution that did not complete its entire flow and was

terminated abruptly. Completing its entire flow, means that the orchestration’s

instance starts in one start activity, and ends in one predicted end activity

Table 2 displays several end activities. Observing non-predicted end activities,

can help to understand where instances failed their execution.

<table>
<thead>
<tr>
<th>End Activities</th>
<th>Scope</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>“RecebeMensagemDadosPassaporte”</td>
<td>“Interpol”</td>
<td>1</td>
</tr>
<tr>
<td>&quot;RecebeMarcacaoLugar&quot;</td>
<td>“MarcacaoLugares”</td>
<td>5</td>
</tr>
<tr>
<td>“EnviaPedidoMarcacaoWebService”</td>
<td>“MarcacaoLugares”</td>
<td>1</td>
</tr>
<tr>
<td>“EnviaMensagemFactura”</td>
<td>“Facturacao”</td>
<td>12</td>
</tr>
</tbody>
</table>

Table 2. Non-predicted end activities retrieved from SportTicket Online’s buying process

Sporticket Online’s buying process was terminated abruptly 19 times, 12 of which were due to failures when calling the billing service, seeing that the activity “EnviaMensagemFactura” is responsible to invoke the billing service. Another 6 failures were due to connections with the stadium management service, since both activities, “EnviaPedidoMarcacaoWebService” and “RecebeMarcacaoLugar” call the stadium management service and receive answers from such service, respectively. One terminated buying process happened while receiving an answer from the Interpol service, which might indicate, either a service failure or a connection problem.

The presented information is useful to broaden orchestration knowledge, either understanding and characterizing correctly executed orchestrations and invoked external services, but also inspecting abruptly terminated orchestration instances and external services. Analysts can therefore quantify and correctly evaluate orchestration execution performance.

Although the presented case study is a brief introduction and sample of the possible analysis that can be performed using the proposed methodology, it has pointed out that Orchinsider’s ADSO mining methodology can help analysts and developers to achieve a higher understanding of the process execution and development, and this way, correctly measure, evaluate and reengineer business processes.

3Last activity executed by one orchestration instance.
7 Conclusion

Using process mining techniques, an orchestration mining methodology was proposed and developed, with the purpose of enabling organizations to analyse the run-time behaviour of service orchestrations, extract executed models and compare them with their original models in order to improve or redesign business processes. The ADSO mining methodology was implemented by a software application, OrchInsider, that was developed to analyse and discover service orchestrations using Microsoft BizTalk integration server. Presently, there is no related work concerning specific service orchestration analysis and discovery, nor similar applications available nowadays, which reinforces this work’s relevance.

Applying process mining techniques to deployed service orchestrations using their run-time generated event logs has delivered useful insights that can be used by organizations to seize knowledge control over their business processes. Understanding how organizations work is better achieved and realized by correctly managing the organization’s business processes, and process mining has proven its usefulness and importance, by analysing and discovering business processes developed, deployed and executed under the form of service orchestrations.

References